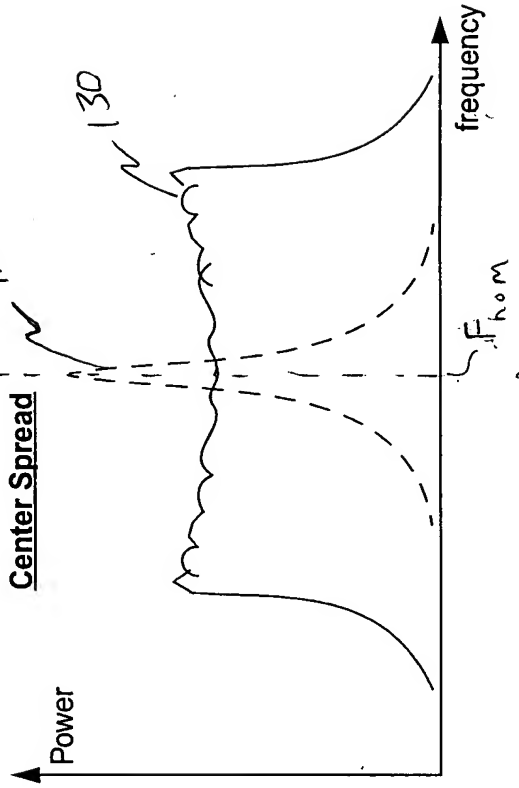
Center Spread
Power
frequency
120
110
130
$F_{nom}$

FIG. 1A

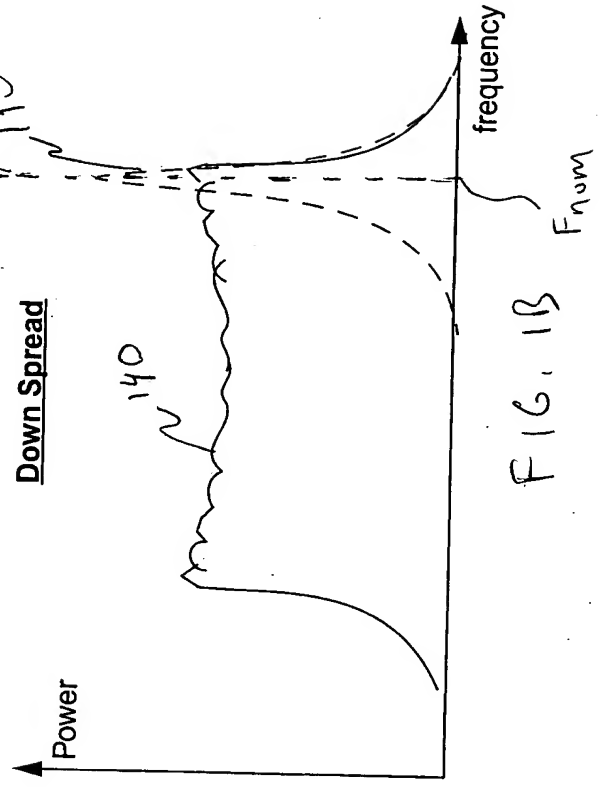
Down Spread
Power
frequency
120
110
140
$F_{nom}$

FIG. 1B

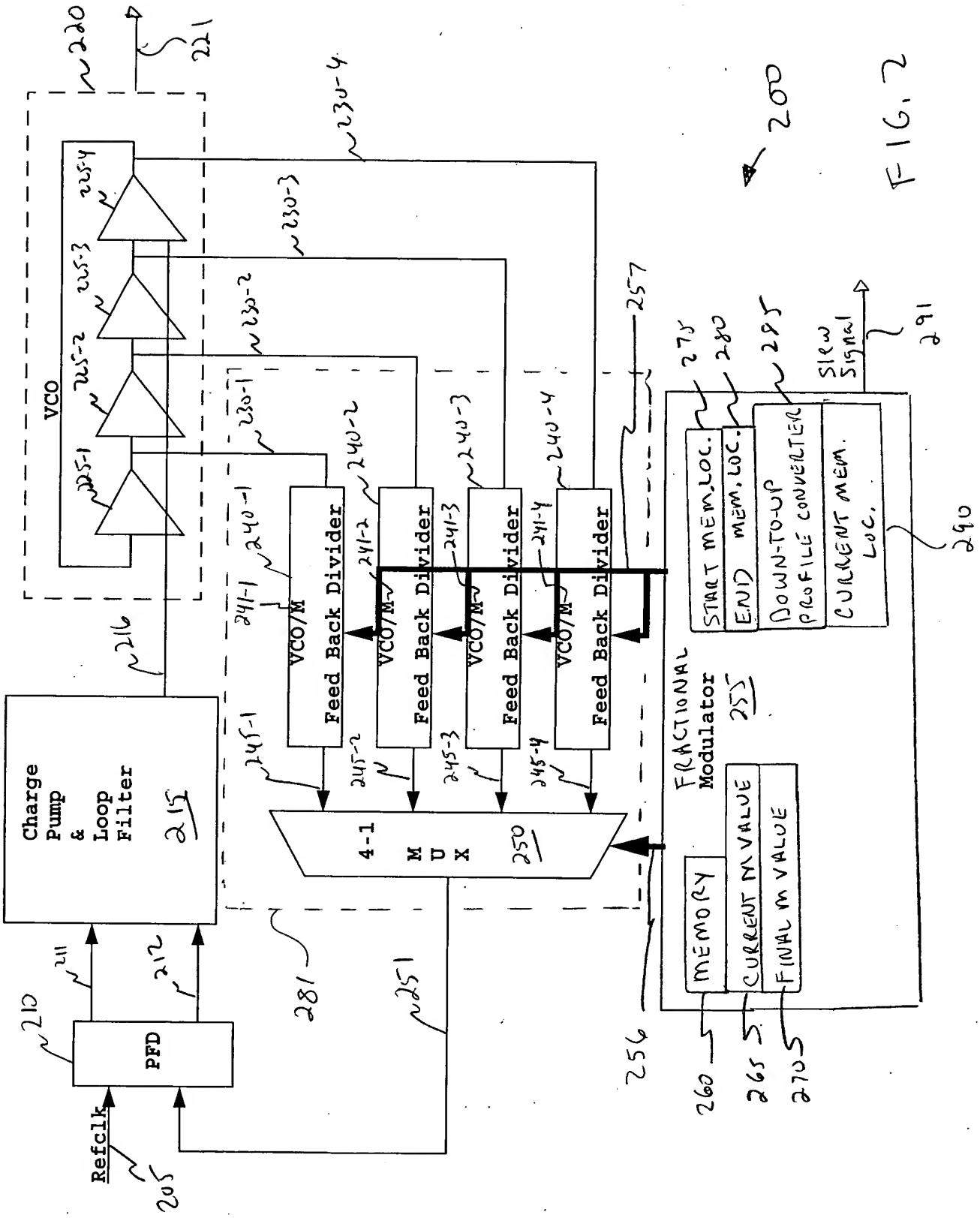
200
VCO
220
221
230-1
230-2
230-3
230-4
235-1
235-2
235-3
235-4
216
Charge Pump & Loop Filter
215
PFD
210
211
212
Refclk
205
241-1
241-2
241-3
241-4
240-1
240-2
240-3
240-4
VCO/M
Feed Back Divider
245-1
245-2
245-3
245-4
4-1 MUX
250
281
251
257
FRACTIONAL Modulator
255
MEMORY
CURRENT M VALUE
FINAL M VALUE
256
260
265
270
START MEM. LOC.
END MEM. LOC.
DOWN-TO-UP PROFILE CONVERTER
CURRENT MEM. LOC.
275
280
285
290
SLEW Signal
291

FIG. 2

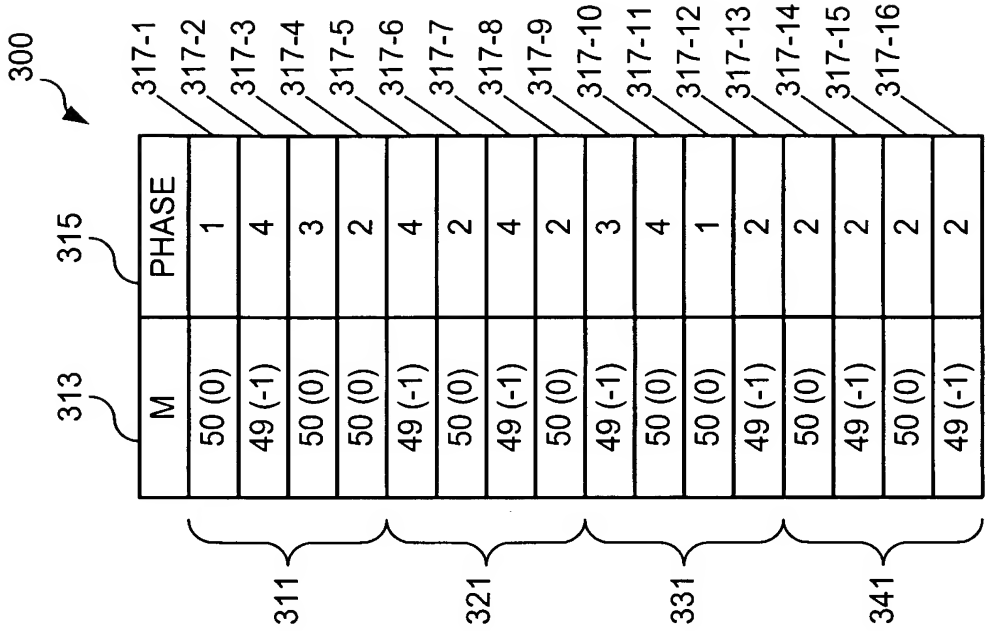
300
317-1
317-2
317-3
317-4
317-5
317-6
317-7
317-8
317-9
317-10
317-11
317-12
317-13
317-14
317-15
317-16
315
PHASE
313
M
1
4
3
2
4
2
4
2
3
4
1
2
2
2
2
2
50 (0)
49 (-1)
50 (0)
50 (0)
49 (-1)
50 (0)
49 (-1)
50 (0)
49 (-1)
50 (0)
50 (0)
49 (-1)
50 (0)
49 (-1)
50 (0)
49 (-1)
311
321
331
341

FIG. 3B

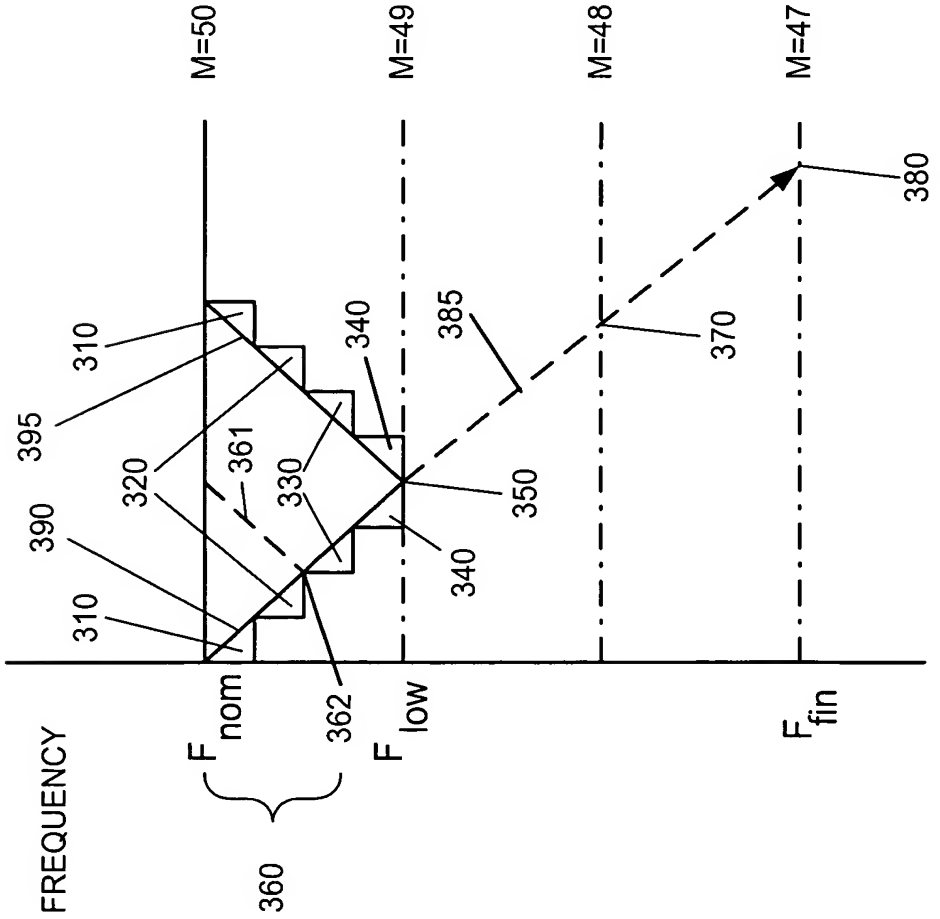
FREQUENCY
M=50
M=49
M=48
M=47
F nom
F low
F fin
360
362
310
310
390
395
320
361
330
340
340
350
385
370
380

FIG. 3A

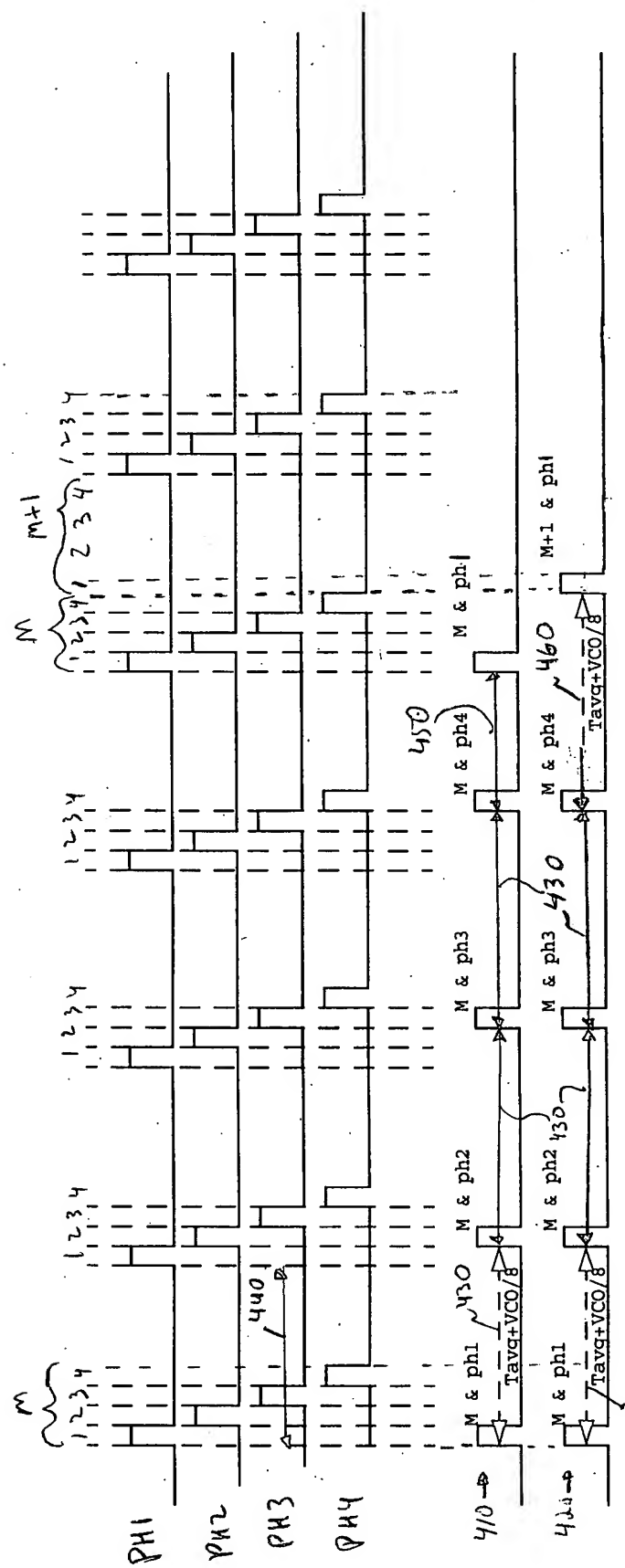
PH1
PH2
PH3
PH4
M
M+1
1 2 3 4
M & ph1
M & ph2
M & ph3
M & ph4
M+1 & ph1
Tavg+VCO/8
410
420
430
440
450
460

FIG. 4


301
1
2
3
4

FIG. 3C

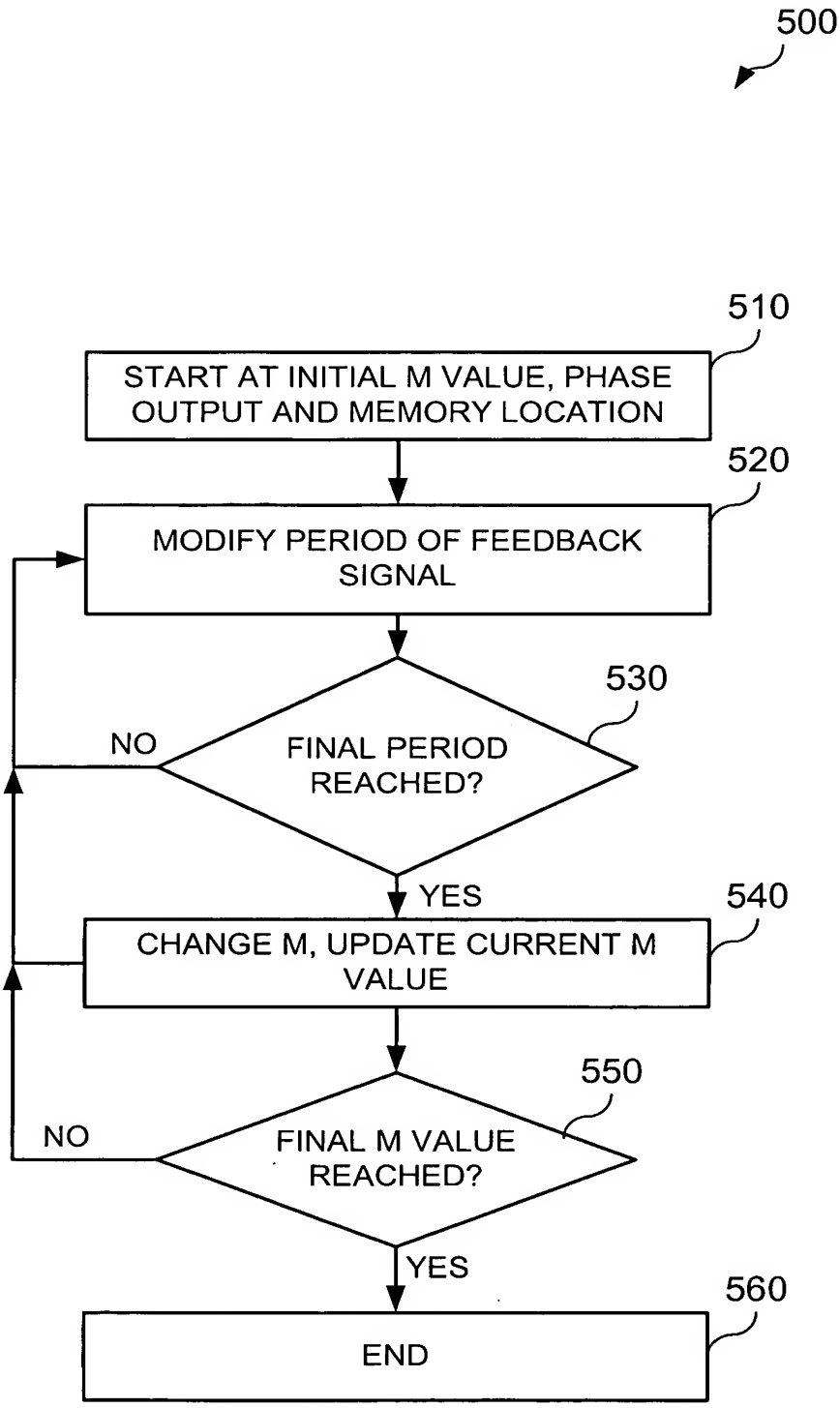
500
510
START AT INITIAL M VALUE, PHASE OUTPUT AND MEMORY LOCATION
520
MODIFY PERIOD OF FEEDBACK SIGNAL
530
FINAL PERIOD REACHED?
NO
YES
540
CHANGE M, UPDATE CURRENT M VALUE
550
FINAL M VALUE REACHED?
NO
YES
560
END

FIG. 5

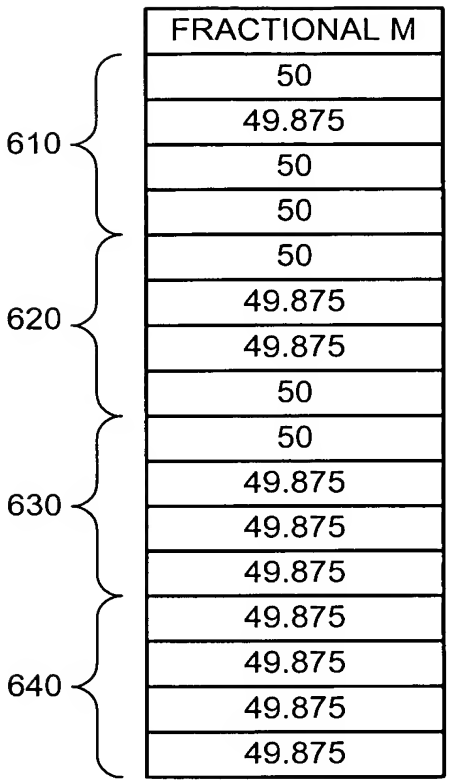
FRACTIONAL M
50
49.875
50
50
50
49.875
49.875
50
50
49.875
49.875
49.875
49.875
49.875
49.875
49.875
610
620
630
640

FIG. 6

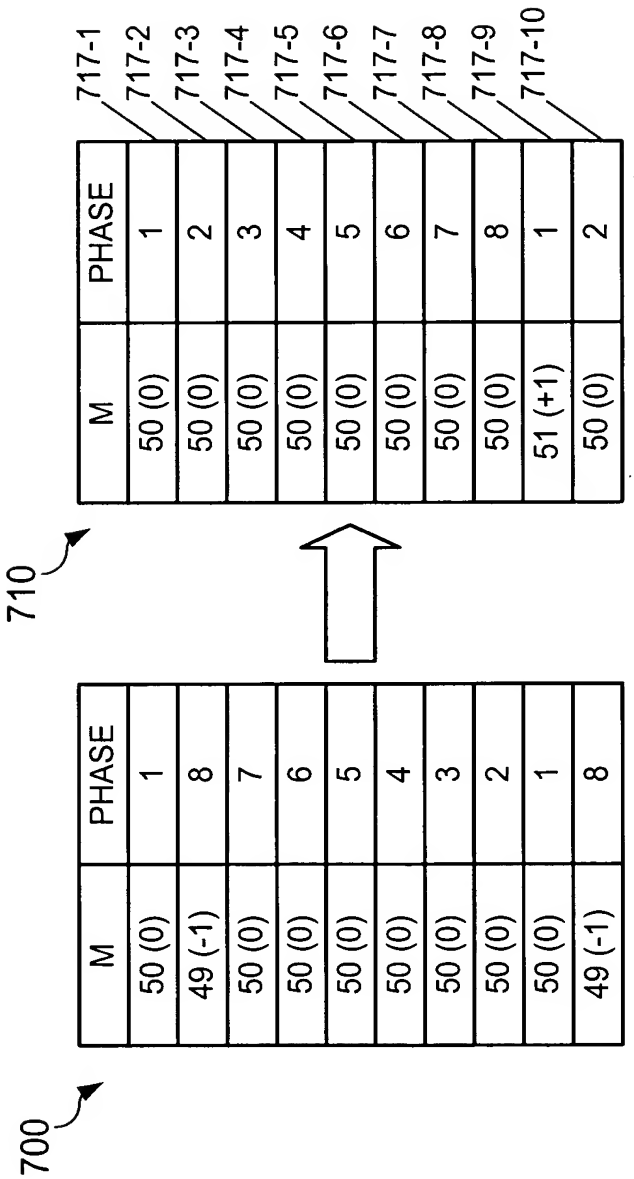
700
M
PHASE
50 (0) 1
49 (-1) 8
50 (0) 7
50 (0) 6
50 (0) 5
50 (0) 4
50 (0) 3
50 (0) 2
50 (0) 1
49 (-1) 8
710
717-1
717-2
717-3
717-4
717-5
717-6
717-7
717-8
717-9
717-10
M
PHASE
50 (0) 1
50 (0) 2
50 (0) 3
50 (0) 4
50 (0) 5
50 (0) 6
50 (0) 7
50 (0) 8
51 (+1) 1
50 (0) 2

FIG. 7

```
module dn_to_up_profile        ( // Inputs
                                 slew_up, Pll_Clk_Early, RESET,
                                 IN_MOFSET2, IN_MOFSET1, IN_MOFSET0,
                                 IN_SELPH3, IN_SELPH2, IN_SELPH1, IN_SELPH0,
                                 // Outputs
                                 OUT_MOFSET2, OUT_MOFSET1, OUT_MOFSET0,
                                 OUT_SELPH3, OUT_SELPH2, OUT_SELPH1, OUT_SELPH0
                               );

input  slew_up;           // Active high control.  Outputs=inputs if slew_up=0.
input  Pll_Clk_Early;     // Clock.
input  RESET;             // Active high reset.
input  IN_MOFSET2;        // The M bits.
input  IN_MOFSET1;
input  IN_MOFSET0;
input  IN_SELPH3;         // Expect always 0 so not sampled.
input  IN_SELPH2;         // The phase bits.
input  IN_SELPH1;
input  IN_SELPH0;

output OUT_MOFSET2;       // The M bits.
output OUT_MOFSET1;
output OUT_MOFSET0;
output OUT_SELPH3;        // Always 0.
output OUT_SELPH2;        // The phase bits.
output OUT_SELPH1;
output OUT_SELPH0;

reg  [2:0] last_selph;
reg  [2:0] local_selph;
wire [2:0] current_selph;


always @( IN_SELPH2 or IN_SELPH1 or IN_SELPH0 )
  begin
     // 2s-complement implementation
     case ( {IN_SELPH2, IN_SELPH1, IN_SELPH0} )
       3'b000 : local_selph = 3'b000;
       3'b001 : local_selph = 3'b111;
       3'b010 : local_selph = 3'b110;
       3'b011 : local_selph = 3'b101;
       3'b100 : local_selph = 3'b100;
       3'b101 : local_selph = 3'b011;
       3'b110 : local_selph = 3'b010;
       3'b111 : local_selph = 3'b001;
     endcase // case( {IN_SELPH2, IN_SELPH1, IN_SELPH0} )
  end // always @ ( IN_SELPH2 or IN_SELPH1 or IN_SELPH0 )


assign     current_selph[2] = slew_up ? local_selph[2] : IN_SELPH2;
assign     current_selph[1] = slew_up ? local_selph[1] : IN_SELPH1;
assign     current_selph[0] = slew_up ? local_selph[0] : IN_SELPH0;


always @( posedge RESET or posedge Pll_Clk_Early)
  begin
       if ( RESET == 1'b1 )
         begin
            last_selph <= 4'b0000;
         end
```

FIG. 8

```
        else
          begin
             last_selph <= current_selph;

          end
    end // always @ ( posedge RESET or posedge Pll_Clk_Early)


assign      OUT_SELPH3 = slew_up ? 1'b0 : IN_SELPH3;
assign      OUT_SELPH2 = slew_up ? local_selph[2] : IN_SELPH2;
assign      OUT_SELPH1 = slew_up ? local_selph[1] : IN_SELPH1;
assign      OUT_SELPH0 = slew_up ? local_selph[0] : IN_SELPH0;

// MOFSET for slewing up will be either 000 or 001, depending
// on whether the phase has just rolled over from 7 to 0.
assign      OUT_MOFSET2 = slew_up ? 1'b0 : IN_MOFSET2;
assign      OUT_MOFSET1 = slew_up ? 1'b0 : IN_MOFSET1;
assign      OUT_MOFSET0 = slew_up ? ( ( (last_selph >  local_selph) ||
                          ((last_selph == local_selph) &&
                           (IN_MOFSET2 == 3'b1) &&
                           (IN_MOFSET1 == 3'b0) &&
                           (IN_MOFSET0 == 3'b1) ) )
                                    ? 1'b1 : 1'b0 ) : IN_MOFSET0;

endmodule
```

FIG. 8 (CONT.)

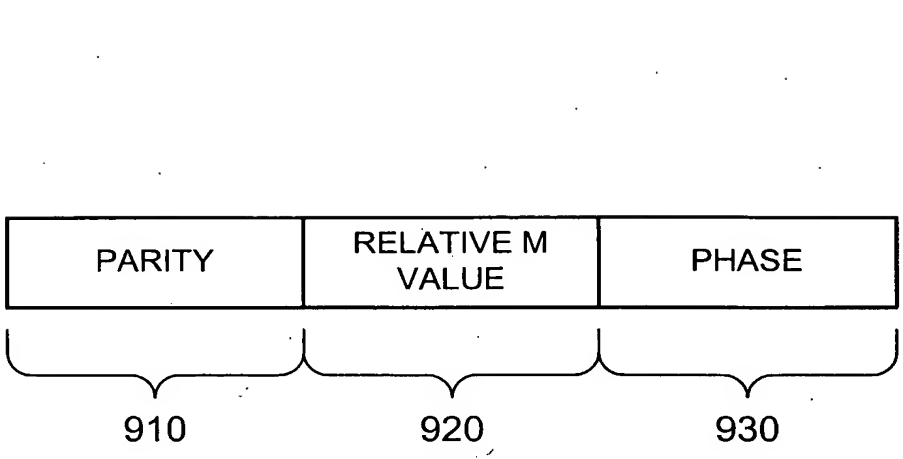


FIG. 9



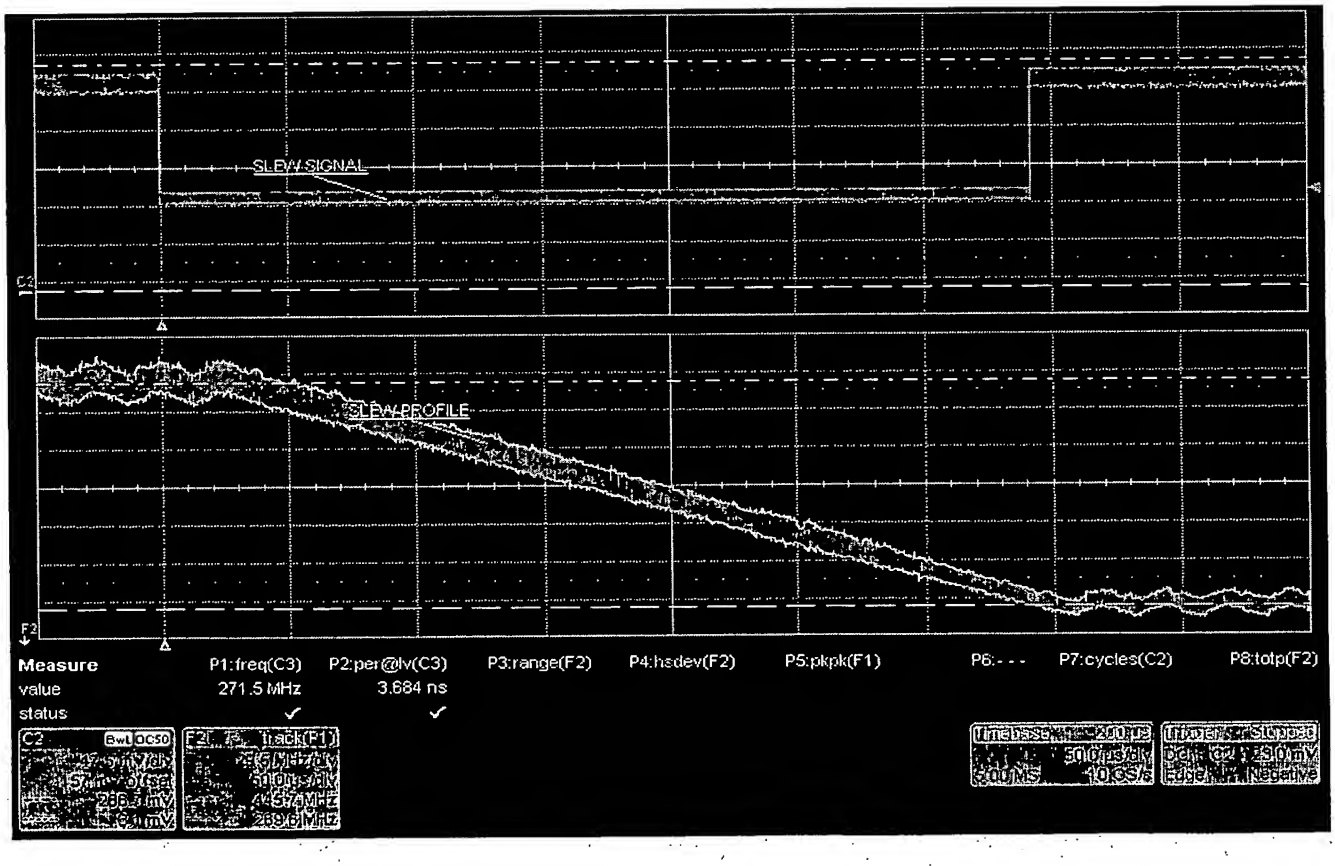


FIG. 10